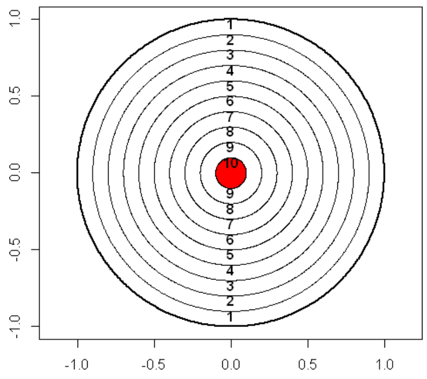
Croce Roberta - Godi Valeria - Zitelli Francesco

## Dart game – CompTools2012 user guide

In this user guide we are going to solve some problems related to the dart game.
Follow us step by step and enjoy the game!
We want to build a target face for throwing darts, like the one in the figure.
We assume that it is a circle of radius 1 with center in the origin point.
For the construction of the target, the reference is paragraph 6.2 of Using R for scientific computing by Karline Soetaert.

We present here the necessary script:

```
> a <- seq(0,2*pi, len=1000)
>plot(cos(a),sin(a),type="l",lwd=2,xlab=""
,ylab="",axes=T, asp=1)
Ten Inner circles at equal distance:
> for (i in seq( 0.1,0.9,by=0.1))
lines(i*sin(a), i*cos(a))
>polygon(sin(a)*0.1,cos(a)*0.1,col="red")
> for (i in 1:10) text(x = 0, y = i/10-
0.025, labels = 11-i, font = 2)
>for (i in 1:9) text(x = 0, y = -1 +
(i/10-0.025), labels = i, font = 2)
```



Basically we have defined a sequence of 1000 angles ranging from 0 to 2π, in radiants; we have plotted them and the result is a circumference. `lines` draws 9 inner circles, with radius equal to 1 * `i`, assuming that the values from 0.1 to 0.9 are at a constant distance of 0.1 (`by=0.1`). By means of the `for` loop, function lines, related to `i`, is repeated for all the values of i, defined in the first line of the for loop in brackets. The same structure is used to assign numbers to circles, through the function `text`.

**Target face and probabilities**

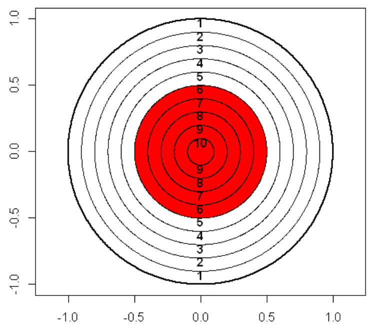
**a) What is the probability of hitting any possible point within the inscribed area of radius ½?**

To draw the inscribed circle, apply a shrinking factor to both `sin(a)` and `cos(a)` in `polygon` function:

```
> polygon(.5*sin(a),.5*cos(a),col="red")
```

The polygon command will cover the inner circles: launch again the previous `for` loops to obtain the following image:

```
> for (i in seq( 0.1,0.9,by=0.1))
lines(i*sin(a), i*cos(a))
> for (i in 1:10) text(x = 0, y = i/10-
0.025, labels = 11-i, font = 2)
>for (i in 1:9) text(x = 0, y = -1 +
(i/10-0.025), labels = i, font = 2)
```



From an analytical point of view, the problem is trivial: it can be solved by computing the ratio of the areas of the two circles.
Thus define the area of a Circle as a function of radius x, through the usual formula $\pi r^2$:

```
> areaC <- function(x) pi*x**2
> areaC(1/2)/areaC(1)
[1] 0.25
```

**b) What is the probability of hitting a point in the first quadrant of the circle of radius 1?**
Again this problem is really simple from the mathematical point of view, since the quadrant is the fourth part of a circle:

```
> areaC(1)/4
[1] 0.25
```

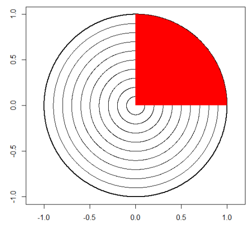
In this case, difficulty stands in drawing the quadrant.
First, draw the basic target face as before and create a new vector, b, from 0 to π/2, that is the angle of the first quadrant:

Circle and the inner ten circles:

```
>plot(cos(a),sin(a),type="l",lwd=2,xlab=""
,ylab="",axes=T, asp=1)
> for (i in seq( 0.1,0.9,by=0.1))
lines(i*sin(a), i*cos(a))
> b <- seq(0, pi/2, len=1000)
```

The command polygon assumes that the polygon is to be closed by joining the last point to the first point. Note that the first two arguments of polygon are respectively the x and y of the figure, and that, in this case, defining b as a sequence of values, polygon joins all the points along the circumference for any value of b, identifying the quadrant.

```
> polygon (c(0,cos(b)),c(0,sin(b)), col=2)
```



The operator c, standing for concatenate, instructs the processor to take together all the arguments inside the brackets.
The result of those code lines is the desired quadrant.

**Be careful!** The polygon function is a low-level command and thus it does not overwrite the previous graphical functions but it is added to the previous by default.

**c) What is the probability of hitting the center (or any other point) in the circle?**
This is a particular case that calls for some theory: the probability of hitting a point in the continuous space is equal to zero. Using our previous function areaC we are able to compute the area of a degenerate circle centered at the origin, that is the center:

```
> areaC(0)/areaC(1)
[1] 0
```

The same could be replicated for any other point considered as the center of a new circle of radius 0.

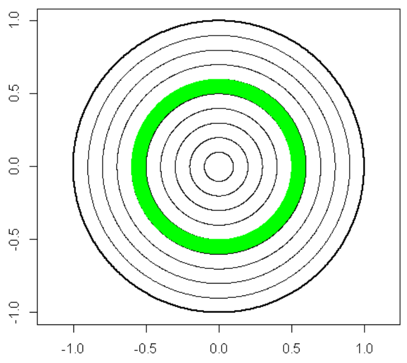
**d) What is the probability of hitting a point in a given annulus (ring), say number 5?**
Once more, the obstacle is given by the representation rather than the computation.
We rebuild the target as we did before:

```
> a <- seq(from = 0, to = 2*pi, len = 100)
> plot(cos(a), sin(a), type = "l", lwd =
2, xlab = "", ylab = "", axes = T, asp= 1)
> for (i in seq(from = 0.1, to = 0.9, by =
0.1)) lines(i*sin(a), i*cos(a))
```

Then we underline the area we are interested about:

```
>polygon(c(sin(a)*0.6,sin(a)*0.5),c(cos(a)
*0.6,cos(a)*0.5),col="green", border =NA)
```

The area is defined for a radius comprised between 0.5 and 0.6. The outcome is represented by the following figure.



Finally we can calculate the probability of hitting the fifth annulus, totalizing a score of 5:

```
> (areaC(0.6)-areaC(0.5))/areaC(1)
[1] 0.11
```

Note that the probability of hitting annulus is increasing in the distance from the center:

```
> (areaC(0.7)-areaC(0.6))/areaC(1)
[1] 0.13
> (areaC(1)-areaC(0.9))/areaC(1)
[1] 0.19
[...]
```

**Empirical tests**

Is theoretical probability the right approximation of reality?

We propose the case of an archer performing 100 throws. All points have the same chance, thus the distribution of his throws is uniform.

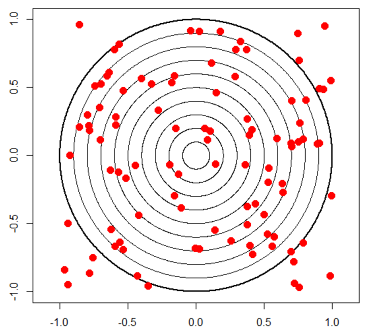
For the structuring of the points, we can think of their coordinates as two separate vectors, each of n elements to be matched according to the respective indices.

```
> x <- runif(100,-1,1)
> y <- runif(100,-1,1)
```

`runif(n,a,b)` stands for *random uniform* and it generates uniform distributed n numbers, ranging from a to b. Note that any time you R launch `runif` function, the variable of destination will change its values.

We can add the shots to the plot with points, with enlarged (`cex=1.5`) shaped dots (`pch=16`):

```
> points(x,y, pch=16, col="red", cex=1.5)
```



Now, the aim is to check whether our previously theoretical probability fits well empirical data.

**a) How many darts have hit the circle of radius ½?**

Note that some darts are outside the target since coordinates are drawn from the square [-1,1]*[-1,1] by the definition of `runif`. Hence, we will need to discard these throws in order to empirically test the previous probabilities.

We want to count all the points (x, y) that are at a distance lower or equal to the radius: by using the logical operators, the processor states when the relation is respected (TRUE) or not (FALSE). By definition, this two Boolean variable have, respectively, value 1 and 0.

Recall that we have defined x and y as a random uniform distribution of 100 points and that our circumference is defined by equation $x^2+y^2=1$; if you take the squared root on both sides and substitute the equality with an inequality, you obtain $\sqrt{x^2+y^2}\leq 1$. Let R substitute coordinates of points and it is easy to know which points belong to the circle.

```
> pt <- sqrt(x**2+y**2)<=1
[1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
```

Summing the values of pt function we obtain the total number of inner points:

```
> innerpt <- sum(sqrt(x**2+y**2)<=1)
[1] 72
```

In our case 72 throws over 100 are accepted. The next step is counting the points within the circle with radius 0.5 and then divide it with the previous result:

```
> rcircle <- sum(sqrt(x**2+y**2)<=0.5)
> rcircle/innerpt
[1] 0.3972603
```

Notice that the empirical result is quite different from the theoretical: this is due to the low number of shots. Let us increase the number of trials to 10000 and repeat the experiment:

```
> x <- runif(10000,-1,1)
> y <- runif(10000,-1,1)
> (innerpt <- sum(sqrt(x**2+y**2)<=1))
[1] 7852
> (rcircle)/innerpt
[1] 0.2426133
```

Note that now the result is a better approximation of our theoretical result, 25%. The next experiments will be performed using the *10000 trials* approach.

**b) How many darts have hit the first quadrant?**

Nothing new on the scenario of R language here, except for the use of the connective &. This is equivalent to the mathematical symbol of intersection.

```
> quad <- sum(x>=0 & y>=0 &
sqrt(x**2+y**2)<=1)
> quad/innerpt
[1] 0.2466887
```

**c) Probability of hitting a point in annulus 5**

```
> ring <- sum(sqrt(x**2+y**2)>=0.5 &
sqrt(x**2+y**2)<=0.6)
> ring/innerpt
[1] 0.1109272
```

**Two archers: Beginner vs Expert**[1]

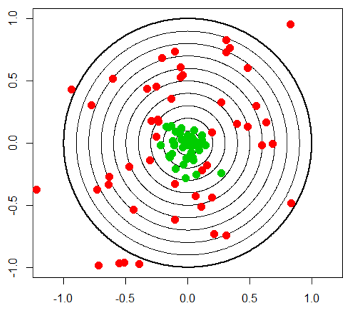
Imagine to have two archers, with different expertise: their shots are normally distributed around the centre but their ability is different and can be mimicked using a different standard deviation. The beginner has a standard deviation greater than the expert, so his shots will be more dispersed than the first:

```
> x <- rnorm(n=100, sd=0.9)
> y <- rnorm(n=100, sd=0.9)
> points(x,y, col="red", pch=16, cex=1.5)
> x <- rnorm(n=100, sd=0.1)
> y <- rnorm(n=100, sd=0.1)
> points(x,y, col="green",pch=16, cex=1.5)
```

Random normal distribution function requires the total amount (n) of trials and the standard deviation (sd). For the sake of brevity, we will show just the first step of our analysis: the following use a similar reasoning of previous examples. The table permits an easy comparison and complete the intuitions we get from the image.

**a) How many beginner's darts have hit the circle with radius 1/2?**

```
> innerpt <- sum(sqrt(x**2+y**2)<=1)
> rcircle <- sum(sqrt(x**2+y**2)<=0.5)
> rcircle/innerpt
[1] 0.05333333
```



| | Beginner | Expert |
|---|---|---|
| Circle (r = 0.5) | 0.05 | 0.33 |
| Quadrant | 0.1 | 0.09 |
| Annulus n. 5 | 0.02 | 0 |

[1] example taken from Soetart's guide